# A Quality Model for Mashups

Cinzia Cappiello[1], Florian Daniel[2], Agnes Koschmider[3], Maristella Matera[1] and Matteo Picozzi[1]

[1] Politecnico di Milano, Dipartimento di Elettronica e Informazione
Via Ponzio 34/5, 20133 Milano, Italy
{cappiell,matera,picozzi}@elet.polimi.it
[2] University of Trento, Dept. of Information Engineering and Computer Science
Via Sommarive 5, 38123 Povo (TN), Italy
daniel@disi.unitn.it
[3] University of Pretoria, Department of Computer Science
0002 Pretoria, South Africa
akoschmider@cs.up.ac.za

**Abstract** Despite several years of mashup practice and research, it is still hard to find high-quality, useful mashups on the Web. While this can be partly ascribed to the low quality of the components used in the mashups or simply to the lack of suitable components, in this paper we argue that this is partly also due to the lack of suitable quality models for mashups themselves, helping developers to focus on the key aspects that affect mashup quality. Although apparently easy, we show that – if taken seriously – mashup development can be non-trivial and that it deserves an investigation that specializes current web quality assessment techniques, which are not able to cater for the specifics of mashups. In fact, we believe a mashup-specific quality model is needed.

## 1 Introduction

*Mashups* are Web applications that integrate heterogeneous resources at different levels of the application stack, i.e., at the data, application logic, and user interface (UI) level. The resources can be readily available on the Web, or they may have been purposely created in order to reach the goal of the mashup developer. For example, it could be necessary to implement a dedicated service to access company-internal data, in order to plot them on a Google map. Independently of where resources come from, the goal of mashups is to provide novel features by integrating resources in a value-adding manner, similar to service composition, which however only focuses on the application logic layer.

The interest in mashups has grown constantly over the last years, especially due to two main reasons. First, mashups and their lightweight composition approaches represent a *new opportunity* for companies to leverage on past investments in service-oriented software architectures and web services, as well as on the huge amount of public APIs available on the Web. In fact, the possibility to integrate web services with UIs finally caters for the development of complete applications, a task that service composition addressed only partially. Second, the

emergence of mashup tools, which aim to support mashup development without the need for programming skills, has moved the focus from developers to *end users*, and from product-oriented software development to consumer-oriented composition [1]. That is, mashup tools have refueled research on and investments in end user development.

While these reasons undoubtedly justify the extraordinary interest in mashups, we however observe that so far none of the envisioned benefits have been achieved. In fact, most of the mashups that can be found online are simple and relatively useless applications and, as such, far from applicable in the enterprise context. Moreover, mashup development is still a prerogative of skilled developers. If we look at the mashups themselves, we observe that their *quality* is simply low, a result that is worsened by the fact that apparently not only end users but even developers have difficulties in implementing high-quality mashups.

Given these premises, the aim of this paper is to understand how to systematically assess the quality of mashups and which are the aspects mashup developers should keep an eye on, in order to obtain mashups of better quality. In our previous research we investigated the quality of *mashup components* [2]. Specifically, we identified a set of quality dimensions and metrics that characterize the quality of mashup components, where each component was seen as a black box, a perspective that characterizes the reuse-centric nature of mashups. Starting from a study conducted on a set of existing mashups, in this paper we define a quality model for *mashups*.

The mashup assessment starts from the assumption that mashups, after all, are Web applications and, as such, can be evaluated by means of *traditional quality models*. We therefore analyzed about 70 mashups available on ProgrammableWeb.com, applying criteria and metrics for Web applications that traditionally focus on the perceived quality of Web applications, e.g., accessibility and usability. This study revealed that understanding the quality of mashups requires a quality model that takes into account the *specifics of mashups*. This led us to the definition of a quality model.

This paper is orgnanized as follows. Next, we review existing techniques and quality models for software and Web application assessment. In Section 3, we apply some of the reviewed tools in order to perform our empirical assessment of mashup quality. In Section 4, we precisely define our idea of mashups and analyze their characteristics from a composition point of view. In Section 5, we derive a quality model that takes into account these characteristics, and we discuss a set of representative positive and negative examples. In Section 6 we conclude the paper.

## 2 Related Work

A quality model is a structured set of quality characteristics that, for instance, allows an assessor to evaluate Web resources from different perspectives. In the Web, first quality models focused on Web sites [3,4], then they focused on more complex Web applications [5,6]. Recently, quality models for Web 2.0 applica-

tions have emerged [7,8]. Yet, the Web 2.0 also introduced a novel role into the quality assessment process, i.e., the end user feedback. In fact, it is nowadays common practice for Web 2.0 applications to be assessed via simple human judgments [9], without following any structured quality model. While user feedback is generally a rich source for quality assessment, the phenomenon of self-ratings by users cannot replace third-party assessments [10,11].

In particular, in the mashup context, traditional quality dimensions suggested for software engineering [12,13] and Web engineering [14,15] may be partly appropriate to measure the internal quality of a mashup (e.g., code readability), as well as its external quality in-use (e.g., usability). In Section 4.3, we will see that, a deeper and more reliable analysis can be achieved by taking into account the component-based nature of mashups. Instead, the W3C guidelines for *accessibility* [16] can be applied to Web sites and mashups alike, as accessibility is a rather technical concern.

Rio and Brito e Abreu [4] found out that most of the quality dimensions are domain-dependent, i.e., that there is a strong impact by the application domain on the usefulness of quality dimensions. In line with this finding, in [2] we started looking at the specifics of mashup components, leaving however open the problem of defining a comprehensive quality model for mashups.

Despite the lack of quality models for mashups, some existing approaches, for instance those focusing on the quality of Web 2.0 information [17,18], may contribute to the assessment of mashups. However, these approaches evaluate the input (resource streams) separately and not their final composition, which is a crucial factor for mashups on which we instead focus. In [19] we tried to partially fill this gap by studying how information quality propagates from components to mashups. Yet, non data-related aspects of internal quality and user interface have not been considered so far in the specific case of mashups.

## 3 Mashup Development: Quality Issues and Challenges

While in the previous sections we generically stated that "mashups are Web applications that integrate heterogeneous resources at different levels of the application stack, i.e., at the data, application logic, and user interface (UI) level," for the purpose of our analysis we further refine this definition as follows:

> Mashups are Web applications that integrate inside one web page *two or more* heterogeneous resources at different levels of the application stack, i.e., at the data, application logic, and UI level, *possibly putting them into communication among each other*.

The first reason for this refinement is that we specifically want to focus on mashups that have an own *UI* (to distinguish them, for example, from so-called data mashups as the ones created with Yahoo! Pipes) and that aim to provide added value by *integrating* a set of existing services or components, rather than coding something from scratch. That is, we want to emphasize the typical *component-based* nature of mashups and the resulting development paradigm of

*composition*. In fact, mashup development is similar to other component-based development paradigms, most prominently to web service composition, which focuses on the integration and *reuse* of existing web services. Mashups, however, extend this composition paradigm also toward data services and UI components, enabling the development of web applications that span all the three layers of the application stack.

If we neglect the static content possibly added by the mashup developer during integration (e.g., in the layout template of the mashup), we can identify two core independent aspects for which *quality* becomes an issue:

- *The components*: Since a mashup reuses data, application functionality, and/or UIs, the quality of these building blocks certainly influences the quality of the final mashup. The lower the quality of the chosen components, the lower the quality of the result of the composition. Even if a developer is aware of the low quality of a component, is it not always possible to choose a better component, e.g., because no other components implementing the same functionality are available.
- *The composition*: While it is usually not possible to improve the quality of third-party components, it is at the other hand relatively easy to further degrade the potential quality of a mashup, i.e., the maximum quality the mashup could have by integrating the same components, if the composition logic of the mashup is not well crafted. In fact, the composition logic is the only part of the mashup that is not reused and that needs to be implemented by the mashup developer, typically each time from scratch. Given the complexity of mashups and the usually high number of different technologies involved, this task is generally error-prone and hard to debug.

Additionally, composition is a complex task, which can be split into three independent sub-tasks, each with its very own quality concerns:

- *Data integration*: Integrating different components may require integrating their data, e.g., if one component is configured to provide input to another component. Doing so may require the re-formatting of data, the cleansing of data, joining or splitting data, and similar.
- *Service orchestration and UI synchronization*: Passing data from one component to another component also means setting up the necessary orchestration logic among services or UI synchronization logic among UI components, since services interactions are typically invocation-based and UIs event-based. Mixing the two integration paradigms is non-trivial in general.
- *Layout*: Finally, one of the most crucial aspects for the success of any application is its graphical appearance. The common practice in mashup development is to use HTML templates with place holders for the UI components to be integrated. Although apparently easy, developing good templates that are able to seamlessly host third-party UIs (possibly with customized CSS settings) is again far from easy.

Given these peculiarities, the *challenge* is to characterize the quality of mashups in a way that (i) captures the perceived quality of the mashup (so as to

address the user's perspective), but that also (ii) allows the mashup developer to act upon the composition logic, in order to settle possible quality issues.

## 4 Assessing Mashups like Common Web Applications

In the following we describe a study that we conducted, in order to understand *how well current mashups perform in terms of existing quality criteria* and *how much useful information a mashup developer can obtain from the application of existing standards and quality evaluation tools for Web applications in general.* The hypothesis of the experiment was that, after all, mashups are nothing but regular Web applications.

We performed a sistematic review of readily available quality assessment instruments, we chose four criteria for which there were automated assessment tools available (in order to eliminate as much as possible subjective evaluations):

- *Usability*: measures the ease of use of the mashup by the mashup users.
- *Readability*: measures how easy or difficult it is to read and understand the text rendered in the mashup.
- *Accessibility*: measures how well the mashup complies with the W3C web accessibility guidelines [16].
- *Performance*: measures the loading time of the mashup till all elements of the application are rendered in the page.

### 4.1 Setup of the study

In order to automatically assess mashups according to the selected criteria, we looked for tools that (i) are able to autonomously access and assess mashups starting from a common URL and (ii) are free. We then identified and selected the following instruments:

- *Site Analyzer* (`http://www.goingup.com/analyzer/`) for the assessment of W3C accessibility and usability guidelines, usability guidelines by J. Nielsen [20] and ISO standards. Altogether, *SiteAnalyzer*$^{TM}$ implements 70 automatically running tests. The evaluation results are shown on a 100-percentage scale, where results equal or higher than 75% indicate a *very good* conformance, result between 75% and 65% a *moderate* conformance and all below 65% represents an insufficient conformance.
- *Juicy Studio* (`http://juicystudio.com/services/readability.php`) for the assessment of readability using the Flesch Reading Ease algorithm, which inspects the average number of words used per sentence and average number of syllables per word in order to compute a number that rates the text on a 100-point scale. The higher the score, the easier the resource to understand. The recommended score for an object is approximately 60 to 70.
- *Pingdom* (`http://www.pingdom.com/`) for measuring mashup loading times. The tool loads all objects (e.g., images, CSS, JavaScripts, RSS, Flash and frames/iframes) and shows the total load time in seconds and visually with time bars.

In order to select the set of candidate mashups for our study, we went to the largest registry of mashups available online, i.e., the ProgrammableWeb.com Web site, which allows mashup developers from all over the world to link their own mashups and to provide some useful meta-data about them. Developer-provided meta-data are, for instance, the *publication date* of the mashup or the set of *APIs* used by the mashups, while the site also features user-provided *ratings* for mashups and the *number of users* that accessed it. Mashups are not hosted on ProgrammableWeb.com, but linked.

Out of the population of 5347 mashups (at the time of the experiment) we randomly selected 224 candidate mashups. We used a *simple random sample* (SRS) technique, a special case of a random sample [21] in which each unit of the population has an equal chance of being selected for the sample. We did not, for instance, focus on the "top 100" mashups, in order to have an as representative as possible sample without bias toward high quality.

Unfortunately, the links of 87 mashups were broken, and 22 mashups could not be processed by the analysis tools. Also, 43 pointed pages could not be considered proper mashups, being them simple Web pages not including any external API. So, the final sample for the evaluation was composed of 68 mashups.

The website of the *Site Analyzer* tool publishes information about the usability and accessibility of top rated web pages. We selected 74 web sites out of 100 (the remaining 26 web sites are variants of each other or duplicates with no difference in rating) and analyzed them according to the selected four quality criteria (usability, readability, accessibility and performance).

Subsequently, we compared the measures achieved for those web sites with the measures of the mashups in order to assess the validity of the values for mashups.

### 4.2 Results

Before the actual quality assessment, we tried to understand whether the meta-data available in ProgrammableWeb.com are somehow correlated to the mashup quality and as such can be considered quality indicators. The performed statistical tests were not able to identify any dependency among the number of users and the average rating of mashups, nor among the publication date and the number of users. This means that the meta-data cannot be used to obtain indications on the quality of mashups.

Table 1, instead, shows the results of the evaluation of the four quality criteria for the five "best" and the five "worst" mashups in terms of usability.

In general, the tool analysis indicates that about 25% of the mashups are of very good usability and accessibility. The average readability degree is close to 60, which indicates an "easy" reading, and the average loading time is close to 5 seconds, which we can consider at the limit of acceptability.

An in-depth analysis of the mashups with very good usability and accessibility (17 mashups) revealed a Spearman's correlation coefficient indicating a high correlation between usability and accessibility (0.855 and p-value $< 0.005$). This means that usability and accessibility are coherent. The average readability

**Table 1.** Evaluation results for the five "best" and five "worst" mashups.

| Rating | Mashup | Usability | Readability | Accessibility | Performance |
|---|---|---|---|---|---|
| 1 | A Paris Hilton video site | 83.9% | 65.2 | 85.5% | 3.1 sec. |
| 2 | Sad Statements | 81.4% | 35.2 | 80.5% | 5.1 sec. |
| 3 | ShareMyRoutes.com | 79.8% | 62.9 | 78.4% | 2.8 sec. |
| 4 | DiveCenters.net | 79.5% | 75.0 | 73.3% | 1.4 sec. |
| 5 | Cursebird | 79.1% | 78.1 | 79.3% | 2.1 sec. |
| ... | ... | ... | ... | ... | ... |
| 64 | CityTagz | 65.0% | 53.3 | 64.2% | 3.2 sec. |
| 65 | Blue Home Finder | 64.9% | 77.7 | 63.9% | 7.3 sec. |
| 66 | Gaiagi Driver - 3D Driving Simulator | 64.8% | 53.9 | 64.6% | 6.3 sec. |
| 67 | 2008 Beijing Olympics Torch Relay Path | 62.2% | 10.5 | 58.7% | 2.0 sec. |
| 68 | Tidespy: Tide Charts with Best Fishing Times | 61.2% | 58.2 | 64.3% | 5.7 sec. |

degree of these 17 mashups was 60.05 and their loading time was 5.33 seconds. In contrast to that, the mashups with very low usability and accessibility had a load time of 3.70 seconds and an average readability degree of 42.66.

Next, in order to understand how mashups with low quality could be improved, i.e., to provide mashup developers with suitable guidelines and to help them improve the mashup, we analyzed the error and warning messages produced by *Site Analyzer*, which specifically focuses on usability and accessibility. In Table 2 we report the five most recurrent warnings and suggestions we collected during our analysis, along with their description.

### 4.3 Analysis of results

In order to understand whether the results of the automatic analysis were reliable, we conducted five independent evaluations by manually inspecting the same mashups and assigning them with a score expressing the mashup quality and a comment justifying the score. We then compared the results of the two evaluation sessions and discovered that in several cases the two assessments diverged.

We immediately identified some inconsistencies. There are indeed some examples that show that the tools were not able to capture some peculiar features of mashups and therefore under- or over-estimated their quality. Let us give an example: Figure 1 shows the Gaiagi 3D Driver mashup, which provides a 3D driving simulation achieved through different map APIs. The user can search for a route, and the application simulates the journey thanks to the Google Earth API and the Google Street View API. A Google Map and a Bing map show a traditional map representation. A text description then shows the route indications. This mashup is very rich from the functionality point of view. Although the included APIs have similar purposes (they all provide map-based services), their orchestration offers a valuable, multi-faceted, still integrated view over route indications. The quality of this mashup, as assessed through the automatic tools, is low: i) the accessibility test obviously failed - a map cannot be interpreted by any accessibility tool (e.g., screen readers); ii) although the usability scores were

**Table 2.** Most recurrent warnings and suggestions for analyzed mashups.

| **Warning** | **%** | **Description** | **Comment** |
|---|---|---|---|
| Visited Links | 68.69 | Marking previously visited hyperlinks helps users to identify which pages of the website have been previously read. In other words, according to Nielsen the navigation interface should answer three essential questions: "Where am I?", "Where have I been?" and "Where can I go? | Mashups typically consist of single pages only. So, the links inside the mashups mostly serve to interact with the single page and its application features, e.g., to show the location of a housing offer on a map, and less to navigate through a complex hypertext structure. Users do not get lost in one-page mashups. |
| Privacy Policy | 64.34 | The owner of the website uses the Privacy Policy to inform users how personal data will be treated and how data protection regulations will be observed. | Given that mashups integrate data from a variety of sources, it may be important to explicitly state where data come from and how they are used. |
| Valid XHTML | 54.78 | Implementing valid (X)HTML markup may improve the quality of the rendering inside the client browser. | Due to their component-based nature, it is hard in mashups to fine-tune the markup code of third-party components. |
| Labelling links on mouse-over | 53.04 | To allow users a good navigation of your website, it is necessary that links can be identified and that they can be highlighted in color with a mouse-over. | Again, this is a features that needs to be implemented by the developers of the components. Adding mouse-over or similar effects are hard to overlay afterwards. |
| Resizable Fonts | 53.04 | Not all visually impaired people make use of technical support to navigate on the internet. The feature to increase fonts is an important element to ensure that all information of the website can be read by all users. Even people who are not visually impaired need to increase the font sometimes. | Increasing or decreasing the fonts inside a mashup may reveal very different behavior from the integrated components. While some of them may implement resizable fonts, others may completely neglect the issue, revealing it only when readability instead needs to be increased. |
| **Suggestion** | **%** | **Description** | **Comment** |
| Sitemap | 90.43 | Sitemaps allow users to have a quick and complete overview of the website. | As already pointed out, mashups typically consist of one page only. So, a sitemap is actually not needed. |
| Printer Friendliness | 84.34 | Web users often want to print the content of a page. Neat printing of the content (e.g., without having the borders cut off) can be achieved by implementing a print function and also by including stylesheets which have been optimized for printing. | While this is a capability that is definitely desirable also for mashups, so far mashups have been focusing more on outputs that are either data (e.g., Yahoo! Pipes) or interactive UIs, which are not easy to print. |
| Table Summary | 61.73 | The TABLE element takes an optional SUMMARY attribute to describe the purpose and/or structure of the table. The overview provided by the SUMMARY attribute is particularly helpful to users of non-visual browsers. | If tables are used for the layout of the mashup, a summary could indeed help impaired users to understand the structure of the mashup. Tables inside the components are outside the control of the mashup developer. |
| Image with Missing Width or Height | 59.13 | Missing height and width size of images impacts the performance of the website. A precise definition of height and width allows a quicker download of the website, as the browser recognizes the space needed for the image and can leave it empty. | This suggestion applies equally to mashups. Given the typically dense integration of contents (components) inside a mashup page, correct formatting may be crucial for successful rendering. |
| Table Header | 46.08 | Table headers should be recognizable as such, since they perform a descriptive task. | The same holds for mashups. Again, tables of the layout template are under the control of the mashup developer, tables inside components not. |

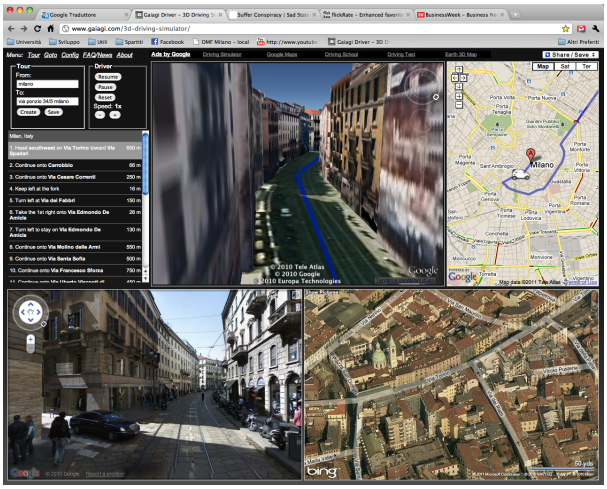

**Figure 1.** A mashup integrating different map-based components (`http://www.gaiagi.com/3d-driving-simulator/`).

moderate (Readability = 53.9; Usability = 64.6%) the mashup was ranked at position 66 out of the 68 mashups.

Conversely, according to the five manual inspections, this mashup was the top ranked: it is easy to use, effective from the point of view of the offered data and functionality, highly interactive and very well orchestrated. This therefore shows that the automatic tools do not capture at all this good quality. One reason is that some criteria that are cornerstone for traditional Web applications, e.g., the richness of links and intra-page navigation, and the readability of text, do not necessarily apply to mashups. For example, map-based components, as the ones used in the example of Figure 1, do not necessarily show text, rather they visually render some points on the map space. Their effectiveness is however still high and proved in several contexts.

Another reason for the observed discrepancy is that, as better discussed in Section 5, important ingredients in mashups are the richness and suitability of components, as well as the way they synchronize. Mashups should indeed be able to offer an integrated view over the different domains deriving from the integrated services. This characterization is not obvious; in other words there is no general consensus on what a mashup is and what not. This is also confirmed by the fact that 29% of the mashups randomly selected from ProgrammableWeb.com included one single API (in most cases a map), without offering any real integration among multiple components. However, we strongly believe that a discriminant factor, which allows one to identify the actual value of a mashup, is the presence of multiple components and a reasonable number of coupling mechanisms giving place to a real integration. These aspects cannot be captured by automatic tools that therefore provide too generic feedback not reflecting at all the salient characteristics of mashups.

The shortcomings of state-of-the-art tools are however not limited to the computation of quality metrics only. In fact, as the comments in Table 2 show, also the feedback provided to developers in terms of warning messages and suggestions is only hardly applicable in the context of mashups. It is generally hard to change any of the hypertext characteristics of the components used inside a mashup, while it is definitely possible to fine-tune the composition logic of the mashup. Yet, this latter aspect is not supported in terms of suitable warnings or suggestions.

## 5 The Mashup Quality Model

As the previous analysis shows, existing quality dimensions are not totally suitable for evaluating the quality of mashups. In this section, we therefore aim at identifying a set of properties that are able to capture the capability of a mashup to enable users to access specific content and functions with easiness, effectiveness, safety, and satisfaction. As we have seen in Section 2, traditional Web pages are usually evaluated by focusing on the value of the provided information and the suitability of the presentation layer [22]. On the basis of the definition provided in Section 3, mashup quality also depends on the quality of the components involved in the mashup and on the quality of the composition in terms of data, functionalities, and interface integration. Therefore, for mashups, traditional dimensions should be revised and enriched with dimensions addressing the added-value that these applications provide through the integration of different components.

We propose a model that classifies quality dimensions into three categories, i.e., *Data Quality*, *Presentation Quality*, and *Composition Quality*, where the latter category is very peculiar for mashups since it focuses on the way components interact among each other and measures the utility of the composition. In order to better define these three categories (in the rest of this section), we first characterize the mashup composition task as follows:

- Mashups are developed in order to offer a set of application features $FS$, and consequently retrieve and give access to a set of data that we call the Data Set $DS$. An *application feature* is a functional capability, i.e., an operation that can be executed by the users or automatically performed on behalf of the users (e.g., due to synchronization between components).
- A mashups can be associated with the set of components $C = \{c_1, ..., c_k\}$ used to create it. Each component $c_i, 1 < i < k$, has its own set of application features $FS_i$ and data set $DS_i$. To fulfill the mashup requirements, smaller portions $SFS_i$ and related $SDS_i$ are sufficient. They are called situational features and data sets.

Figure 2 sketches the quality aspects and dimensions addressed by our mashup quality model, which we discuss next in more detail.

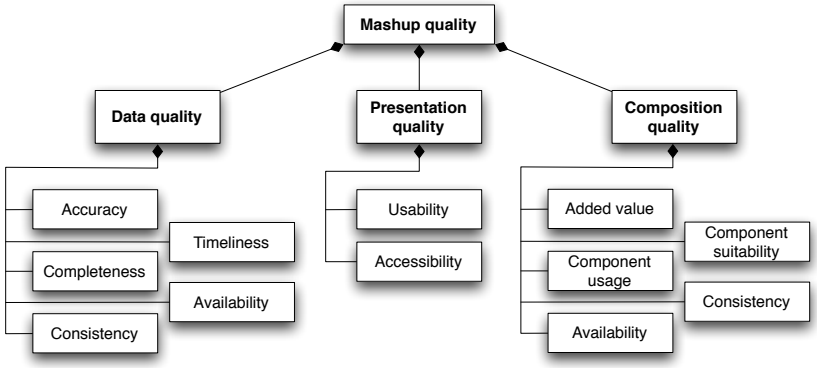


**Figure 2.** The dimensions of the quality model for mashups.

### 5.1 Data quality

As defined in [19], the mashup quality strongly depends on the information that the integration of different components is able to provide. Assessing the quality of a mashup therefore requires understanding both the quality of components and the effect that the composition has on the overall quality of the final mashup. In [19], we have defined an information quality model for mashups from which we reuse the following set of dimensions relevant for mashup applications:

- *Accuracy*: refers to the probability that mashup data are correct. This is calculated by considering the probability that an error occurs in the data sets $SDS_1$,...,$SDS_k$ of the components used in the mashup. This probability can be estimated by looking at the usage history of a component and considering all the types of occurred errors (e.g., typos, wrong representation).
- *Situational Completeness*: defines the degree with which $DS$ is able to provide the desired amount of information. The evaluation of such dimension starts from considering the amount of data contained in $SDS_1$,...,$SDS_k$ and comparing them to the amount of data that ideally should be accessed through the mashup.
- *Timeliness*: provides information about the freshness of the available data sets. The timeliness of the mashup results from the aggregation of the timeliness of the different components where the aggregation function can be minimum, average, or maximum.
- *Consistency*: addresses situations in which the mashed up components have data sets that conflict with each other, leading to inconsistency in the data shown in the final mashup.
- *Availability*: is the likelihood for the mashup to be able to provide any data, that is, in order for a mashup to be available at least one of its components must be available.

It is worth to point out that the way in which components interact among them influences the information quality assessment, especially for accuracy and sit-

uational completeness dimensions. Timeliness and availability are instead computed as simple aggregation of the respective characteristics of the individual component data sets.

### 5.2 Presentation quality

*Presentation Quality* is traditionally decomposed into *usability* and *accessibility*:

- *Usability*: also in the mashup context, usability addresses some traditional dimensions, such as *orientation*, *users control*, *predictability*, *layout consistency*. For example, the features provided by a mashup should be invoked through commands that are easy to identify and understand. The users should also easily understand the effect of a feature when it is invoked. In particular, in a mashup that synchronizes multiple components, the effects of the propagation of a command over the different components should be visible to the user.

  An important usability attribute is *learnability*, which relates to"the features of an interactive system that allow novice users to use it initially, and then to attain a maximum level of performance" [23]. Learnability should be privileged in mashups, even though this application are very often simple: the mashup features should be visible enough and the corresponding commands should be self-expressive so that even naive users can easily master the mashup execution.

  Finally, a major role is played by *layout consistency*, which provides the "glue" through which even heterogenous components result to be a whole. However, some other usability attributes usually adopted for traditional Web applications are not applicable in the evaluation of mashup quality. First of all, even though future development practices would lead to the creation of multi-page mashups [24], current mashups are still very simple applications from the point of view of their hypertext structure – very often they are made of a single page. Therefore, criteria such as navigability and richness of links, or any other criteria addressing the hypertext structure, which are generally considered relevant for the quality of Web applications [25], have a lower impact over the mashup quality.

  Moreover, mashup applications rarely contain long texts and the assessment of all the quality dimensions that are commonly used for the evaluation of Web content, such as *readability*, *cohesion* or *coherence*, might provide irrelevant results.
- *Accessibility*: it addresses properties of graphic page design, such as those defined by the W3C Web Accessibility Initiative (WAI) [26]. Accessibility criteria do not need to be specialized for mashups. Rules for enabling access by any user and any device should indeed be taken into account when defining the layout of the mashup, also smoothing some layout setting of single components in case they violate any accessibility rule.

### 5.3 Composition quality

As stated above, composition quality aims at evaluating the component orchestration and the suitability of the mashup for the provision of the desired features. The composition quality is decomposed into *added value*, *component suitability*, *component usage*, *consistency* and *mashup availability*.

- *Added value*: the *added value* of the composition can be related to the amount of provided features and/or offered data. The composition provides an added value if, for each component $c_i$, $SFS_i \subset FS \vee SDS_i \subset DS$, i.e., the amount of features/data offered by the mashup is greater than the amount of features/data offered by the single components. While the previous one defines the minimum condition ensuring that some added value is provided by the composition, the goal to be reached is that $\bigcup_i SFS_i \subset FS \vee \bigcup_i SDS_i \subset DS$. This also reflects the principle that information becomes more valuable when it can be compared and combined with other information [27].
  We can quantify the added value along a scale that ranges from the case in which a mashup gives simply the opportunity to render data coming from different sources (without any attempt to integrate them) to the case in which additional features and data are provided by an adequate integration. For example, a mashup as *dailymashup.com* provides a low added value since its single page offers a very poorly integrated view on some selected news (taken from Yahoo!news) and, in a totally unaligned fashion, also on the top-ranked last 24 hours photos from Flickr. More added value would be offered if the mashup components were synchronized. To provide an added value, mashups must offer to the users additional features or data, as for example the mashup published on *www.bluehomefinder.com*, where an advanced service for finding houses offers the localization of houses on a map plus other features that allow the users to perform operations of filtering and selection.
- *Component suitability*: it refers to the appropriateness of the component features and data with respect to the goal that the mashup is supposed to support. For example, a mashup that aims at providing addresses of the nearby restaurants with respect to the current user location should be effectively built based on map-based components. In fact, a simple list of textual addresses could not appropriately support those users that are not acquainted with the geographical area.
- *Component usage*: it may happen that, even though a component is very rich from the point of view of data and functionality, it is improperly used within a composition. For example, the Google Maps API offers several operations, such as geocoding, directions, street view, and elevation. Let us consider that a mashup developer decides to just use the street view feature. This choice is not reasonable if the user goal is to get oriented within a geographical area: the street view just offers a local and realistic view of a selected point of interest, while it does not provide a larger view of the area in which the point is located.
- *Consistency*: poor quality compositions can also be caused by *inconsistencies* at the orchestration level. In fact, the composition of two components

is feasible if the two linked operations are compatible from only a syntactic perspective even if they are incompatible from a semantic perspective. In this way, the composition can produce inaccurate results. For example, the composition between an operation that provides an address and a map service is feasible since the input-output parameters are strings. However, if the address is incomplete and contains only information about the city in which the point of interest is located, this will work properly but it will not show the desired information.
- *Availability*: the degree in which the mashup can be properly accessed during a given time interval. It depends on the availability of the components and on their role in the composition.

## 6 Conclusion and Future Work

To the best of our knowledge, this paper represents the first attempt to define a quality model for mashups. Although many so-called "quality-aware" or "quality-based" composition approaches (mostly for web service composition) have been proposed so far, a holistic approach to quality assessment of the final output of the composition task – especially for mashups – was still missing.

Driven by the results of our study, where we evaluated a huge sample of mashups, we took a better look at the peculiarities of mashups and understood that, after all, mashup developers do not have full control over *what* they integrate (the components) but only over *how* they integrate (the composition logic). Accordingly, we defined a quality model that emphasizes this component-based nature of mashups and especially focuses on composition quality.

We recognize that many of the quality dimensions introduced in this paper are not easy to turn into operative metrics and to automatically assess, yet we also recognize that quality assessment to a large degree will always be a *qualitative* process. In [19] we started proposing more concrete metrics for data quality; usability and accessibility are already supported to a large extent with (semi-) automated evaluation metrics; the next challenge is supporting composition quality with suitable metrics. This is part of our future research.

**Acknowledgments.** This work was partially supported by funds from the European Commission (project OMELETTE, contract no. 257635).

## References

1. Nestler, T.: Towards a mashup-driven end-user programming of soa-based applications. In: iiWAS, ACM (2008) 551–554
2. Cappiello, C., Daniel, F., Matera, M.: A quality model for mashup components. In: Proc. of ICWE. (2009) 236–250
3. Ivory, M.Y., Megraw, R.: Evolution of web site design patterns. ACM Transactions on Information Systems **23** (2005) 463–497
4. Rio, A., e Abreu, F.B.: Websites quality: Does it depend on the application domain? In: Proc. of QUATIC, IEEE Computer Society (2010) 493–498

5. Olsina, L., Rossi, G.: Measuring web application quality with webqem. IEEE Multimedia **9** (2002) 20–29
6. Mavromoustakos, S., Andreou, A.S.: WAQE: a web application quality evaluation model. Int. J. Web Eng. Technol. **3** (2007) 96–120
7. Olsina, L., Sassano, R., Mich, L.: Specifying quality requirements for the web 2.0 applications. In: Proc. of IWWOST 2008. (2008) 50–56
8. T, T.K., Mlýnková, I.: Quality assessment social networks: A novel approach for assessing the quality of information on the web. In: Proc. of QDB, ACM press (2010) 1–10
9. Varlamis, I.: Quality of content in web 2.0 applications. In: Proc. of KES, Springer-Verlag (2010) 33–42
10. Brooks, C.H., Montanez, N.: Improved Annotation of the Blogosphere via Autotagging. In: Proc. of WWW, Edinburgh, UK, ACM Press (2006) 625–632
11. Heymann, P., Koutrika, G., Garcia-Molina, H.: Can Social Bookmarking Improve Web Search? In: Proc. of the WSDM, ACM Press (2008) 195–206
12. Kan, S.H.: Metrics and Models in Software Quality Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
13. ISO/IEC: ISO/IEC 9126-1 Software Engineering. Product Quality - Part 1: Quality model. (2001)
14. Calero, C., Ruiz, J., Piattini, M.: Classifying web metrics using the web quality model. Online Information Review **29** (2005) 227–248
15. Olsina, L., Covella, G., Rossi, G.: Web quality. In Mendes, E., Mosley, N., eds.: Web Engineering: Theory and Practice of Metrics and Measurement for Web Development. Springer (2005) 109–142
16. W3C: Web Content Accessibility Guidelines (WCAG) 2.0. (2008)
17. Abramowicz, W., Hofman, R., Suryn, W., Zyskowski, D.: Square based web services quality model. Information Systems Journal **I** (2008) 1–9
18. Zhang, H., Zhao, Z., Sivasothy, S., Huang, C., Crespi, N.: Quality-assured and sociality-enriched multimedia mobile mashup. EURASIP J. Wirel. Commun. Netw. **2010** (2010) 11:1–11:13
19. Cappiello, C., Daniel, F., Matera, M., Pautasso, C.: Information quality in mashups. IEEE Internet Computing **14** (2010) 14–22
20. Nielsen, J.: The usability engineering life cycle. Computer **25** (1992) 12–22
21. Berger, V.W., Zhang, J.: Simple Random Sampling. Encyclopedia of Statistics in Behavioral Science (2005)
22. Aladwani, A.M., Palvia, P.C.: Developing and validating an instrument for measuring user-perceived web quality. Inf. Manage. **39** (2002) 467–476
23. Dix, A., Finlay, J., Abowd, G.D., Beale, R.: Human Computer Interaction. 3. edn. Pearson, Harlow, England (2003)
24. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From people to services to ui: Distributed orchestration of user interfaces. In: BPM. (2010) 310–326
25. Ceri, S., Matera, M., Rizzo, F., Demaldé, V.: Designing data-intensive web applications for content accessibility using web marts. Commun. ACM **50** (2007) 55–61
26. Consortium, W.: Wai guidelines and techniques. Technical report, `http://www.w3.org/WAI/guid-tech.html` (2007)
27. Measuring The Value Of Information: An Asset Valuation Approach. In: Proc. of ECIS 1999. (1999)